AMENDMENTS TO THE SPECIFICATION

Please amend the title of the invention to read as follows:

"SYSTEM AND METHOD FOR DYNAMIC BUFFER ALLOCATION"

Please amend the remainder of the specification as follows:

On page 2, lines 12-16:

As an example, current Intel[[7]]® Pentium[[7]]® Pro-based personal computers have a 200 MHz processor bus and a 33 MHz Peripheral Component Interconnect (PCI) I/O bus. Due to the speed differential between the Pentium[[7]]® Pro processor bus and the PCI bus, the Pentium[[7]]® Pro processor is forced, in many instances, to wait through several clock cycles before accessing the PCI bus to send address or data information to a peripheral device.

On page 2, lines 17-25:

To circumvent this problem, others have placed First In/First Out (FIFO) buffers between the Pentium[[7]]® processor bus and the PCI bus. For example, the Intel[[7]]® 82433LX Local Bus Accelerator Chip includes a four double word deep processor-to-PCI posted write buffer for buffering data writes from the Pentium[[7]]® processor to peripheral devices on the PCI bus. This buffer is a simple first-in/first-out (FIFO) buffer wherein the destination address is stored in the buffer with each double word of data. In use, the processor-to-PCI posted write buffer is positioned within a bridge circuit, between the processor bus and the PCI bus. As the processor generates data writes to the PCI bus, the data is queued in the posted write FIFO buffer of the Intel[[7]]® 82433LX.

On page 2, lines 26-32:

The FIFO buffered bridge structure of the Intel[[7]]® 82433LX allows the Pentium[[7]]® Pro Processor to complete processor to PCI double word memory writes in three processor clocks (with one wait-state), even if the PCI bus is busy on the first clock. Once the PCI bus becomes available, the posted write data stored in the FIFO buffer is written to the designated PCI device. Uncoupling the processor request from the PCI bus in this manner allows the processor to continue processing instructions while waiting to retrieve the designated information from the PCI bus.

On page 3, lines 1-5:

In addition to the four double word deep posted write buffer, the Intel[[7]]® 82433LX also includes a processor-to-PCI read pre-fetch buffer. The pre-fetch buffer is four double words deep and enables faster sequential Pentium[[7]]® □ Pro Processor reads from the PCI bus. The Intel[[7]]® 82433LX read pre-fetch buffer is organized as a simple FIFO buffer that only supports sequential reads from the PCI bus.

On page 3, lines 6-9:

In practice, data is sent from the PCI bus, through the processor-to-PCI read pre-fetch buffer, to the processor. Processors such as the Intel[[7]]® Pentium[[7]]® Pro include an instruction pre-fetch circuit so they can gather instructions that are about to be executed by the processor.

On page 3, lines 10-15:

Unfortunately, attempts at solving the problem of processors running faster than bus substructures have not met with complete success. Many Intel[[7]]® Pentium[[7]]® Pro-based computer systems that employ FIFO buffering schemes to manage data traffic between the PCI bus and the processor are still inserting one or more wait states into their bus read and write instructions. This lowers the computer system's performance and causes many software programs to run slower than necessary.

On page 3, lines 16-25:

As one example, the Intel[[7]]® 82433LX only provides a limited flexibility for handling data writes and reads to the PCI bus. In particular, the processor-to-PCI posted write buffer and processor-to-PCI read pre-fetch buffer are both unidirectional FIFOs and therefore do not allow for random access of their contents. Moreover, if the processor is performing a tremendous number of write instructions to the PCI bus, the posted write buffer does not have the flexibility to handle more than four double words. Thus, wait states are inserted into the processor clock until the FIFO buffers are cleared. For all of the above reasons, it would be advantageous to provide a system that had the flexibility to allow additional buffers to become available during peak write and read periods. This flexibility is offered by the system of the present invention.

On page 4, lines 1-7:

In one embodiment, a dynamic buffer allocation (DBA) system is located within an Intel[[7]]® Pentium[[7]]® Pro processor to PCI bus bridge circuit. The DBA system may provide a matched set of three address and three data buffers. These buffers act together to manage data flow between the processor and the PCI bus. In addition, the address and data buffers are [[A]]"matched"[[≡]] in the sense that each address buffer works in conjunction with only one particular data buffer. These buffers, as described below, allow for a flexible, bi-directional data flow between the processor and peripheral bus of a computer.

On page 4, lines 26-30:

In another embodiment, the processor is an Intel[[7]]® Pentium[[7]]® Pro microprocessor and the peripheral bus is a Peripheral Component Interconnect (PCI) bus. In such a computer, there are five possible data paths which manage three types of data transfers. The three types of data transfers in the Pentium[[7]]® Pro system are: 1) processor to PCI Write Data, 2) processor to PCI Read Data, and 3) processor to PCI Deferred Data.

On page 4, lines 31-32 to page 5, lines 1-5:

As is known, the Intel Pentium[[7]]® Pro processor may perform a [[A]]"deferred"[[≡]] data read from the PCI bus by setting a transfer bit that accompanies the address request. After the data is read from the PCI device, it is sent to a deferred data handling circuit before being sent to the processor bus. The deferred data handler keeps track of the outstanding deferred data reads and notifies the Pentium[[7]]® Pro processor when a deferred data read from a PCI device is available. Five possible data paths for handling address and data transfers within the DBA system are listed below.

On page 5, lines 24-32:

Another embodiment is a dynamic buffer system in an Intel Pentium[[7]]® Pro computer system for controlling data flow between an Intel Pentium[[7]]® Pro processor and a Peripheral Component Interconnect (PCI) device. The dynamic buffer system includes: a first buffer in communication with a Pentium[[7]]® Pro processor and a PCI device; a second buffer in communication with the Pentium[[7]]® Pro processor, the PCI device and the first buffer; control logic for controlling the first buffer and the second

buffer as matched pairs so that an address stored in said first buffer corresponds to data stored in the second buffer; and an arbiter for controlling bi-directional data flow between the Pentium[[7]]® Pro processor and the PCI device, wherein the data is buffered by the second buffer.

On page 8, lines 1-8:

Figure 1 is a block diagram of a computer system 5. The computer system 5 includes a processor 7 that connects via an external processor bus 9 to a bridge circuit 11. In one embodiment, the processor is an Intel[[7]]® Pentium[[7]]® Pro processor, although other processors can be used in conjunction with the DBA system. Such processors include the Pentium II processor from Intel, the Alpha[[7]]® processor from Digital Equipment Corporation and the PowerPC[[7]]® processor from Motorola Corporation. Integral to the bridge circuit 11 is a dynamic buffer allocation system 13. Within the dynamic buffer allocation system 13 are address and data buffers 15.

On page 9, lines 13-18:

Referring now to Figure 3, a detailed block diagram of one embodiment of a Pentium processor to PCI bridge circuit 40 is shown. An Intel Pentium[[7]]® Pro processor 41 is linked through an external processor bus 42 to the bridge circuit 40. The bridge circuit 40 communicates across a PCI bus 43 to a set of PCI devices 44a-c. Thus, address and data information that is sent to PCI devices 44a-c from the Pentium[[7]]® Pro Processor 42 first passes through the bridge circuit 40.

On page 9, lines 19-25:

As shown in Figure 3, the external Pentium[[7]]® Pro processor bus 42 is in communication with an internal processor bus 45 in the bridge circuit 30. The internal processor bus 45 transfers all address and data communication from the external Pentium[[7]]® Pro bus 42 to the internal components of the bridge circuit 40. Similar to the external Pentium[[7]]® Pro bus 42, in one embodiment the internal processor bus 45 has a 32 bit address bus and 64 bit data bus to manage address and data communication to and from the Pentium[[7]]® Pro processor 41.

On page 9, lines 26-31 to page 10, lines 1-2:

Connected to the internal processor bus 45 is a processor bus master controller 50 and processor bus slave controller 55. The processor bus master controller 50 handles transfers of deferred cycle retries and replies that are sent from the PCI devices 44a-c to the Pentium[[7]]® Pro processor 41. As discussed above, the deferred data is managed by a deferred response handler within the processor bus master controller 50. For a complete discussion of a deferred data handler within a Pentium[[7]]® Pro computer system see Intel[[7]]® Corporation's Pentium Pro Family Developer's Manual, Volume #1 which is incorporated by reference.

On page 10, lines 3-6:

The processor bus slave controller 55 controls address and data writes from the Pentium[[7]]® Pro processor 41 to the bridge circuit 40 and also decodes and directs processor requests to the PCI bus 43. In addition, the processor bus slave controller 55 transfers read data from the designated PCI device 44 to the Pentium[[7]]® Pro Processor 41.

On page 10, lines 7-17:

Linked to the processor bus master controller 50 and bus slave controller 55 is a PCI Master Controller 57 which includes one embodiment of a DBA system 60. As discussed above, the DBA system 60 buffers request and data transfers between the Pentium[[7]]® Pro processor 42 and all of the PCI devices 44a-c residing on the PCI bus 43. CPU requests that are directed to the PCI bus will pass through the PCI Master Controller 57. Other CPU requests will be directed to their correct destination by the CPU Bus Slave Controller 55 or an alternative controller (not shown). The only cycles that the

PCI target controller 62 processes are the cycles generated by the PCI devices 44a-c. The PCI target controller 62 handles requests originated from the PCI devices 44a-c to the processor 41 that route through the bus master controller 50. In addition, the PCI target controller 62 manages PCI device requests that are sent to the main memory of the computer system.

On page 10, lines 18-26:

In order for the bridge circuit 40 to communicate with the external PCI bus 43, an internal PCI bus 65 is provided to place data and address information onto the 32-bit address and 32/64-bit data lines of the external PCI bus 43. Thus, a 32-bit deferred read request to the PCI bus 43 from the Pentium[[7]]® Pro processor 41 travels through the external Pentium[[7]]® Pro bus 42 and onto the internal processor bus 45 of the bridge circuit 40. The bus slave controller 55 decodes the PCI read request and directs it to the DBA system 60. The PCI address that is sent with the Pentium[[7]]® request is then buffered in one of the address buffers (not shown) within the DBA system 60. At this point, the Pentium[[7]]® Pro Processor 41 can continue to execute instructions.

On page 10, lines 27-31:

Once the PCI bus 43 is free to accept read requests from an address buffer within the DBA system 60, the request is sent out along the internal PCI bus 65 and finally outside of the bridge circuit 40 to the external PCI bus 43. From the external PCI bus 43 the read request is sent to a target PCI device 44a-c which accepts the address request and prepares the requested data for transmission to the Pentium[[7]]® Pro Processor 41.

On page 11, lines 1-7:

The requested data follows the opposite path, through the PCI bus 43, internal PCI bus 65 and into a data buffer (not shown) within the DBA system 60. The DBA system 60 then makes a request of the bus master controller 50 to perform a deferred retry or deferred reply cycle to the Pentium[[7]]® Pro processor 41. After the bridge circuit 40 is notified that the processor bus is free, the data is written out to the bus master controller 50 and thereafter placed on the internal processor bus 45, external Pentium[[7]]® Pro bus 42 and finally sent to the Pentium[[7]]® Pro processor 41 for processing.

On page 12, lines 8-15 (indents corrected):

The buffer valid bit is a bit that may be set when an address request initiator, such as the Pentium[[7]]® Pro Processor 41 or PCI device 44, requests a transfer and it is accepted. A cycle initiated by a PCI device 44 is normally sent to the PCI target controller 62 or to another PCI device. The bit may be cleared upon completion of the cycle, indicating that the buffer is available for another address request. This bit may be set when a processor to PCI read or write cycle is initiated by the processor and may be cleared upon the write completing on the PCI bus or the read completing on the processor bus.

On page 13, lines 1-9 (indents corrected):

This bit may be set when the response agent (e.g.: target of the address request) needs to take action. It can be cleared when the response agent is finished performing its task. This bit may be set, for example, when the Pentium processor 41 has written data to the matched data buffer for a processor-to-PCI write cycle and cleared when the data has been written from the data buffer to the PCI bus. In addition, this bit may be set immediately for a processor-to-PCI read and cleared when read data has been returned from the PCI bus to the appropriate data buffer.

On page 14, lines 5-14 to page 15, lines 1-8 (indents corrected):

Transfer Type Bit 0: (Processor Write)

This bit may be set when the processor initiates a write and is cleared when the processor has finished writing data to the data buffer.

Transfer Type Bit 1: (PCI Write)

This bit may be set when an initiator requests a PCI write cycle and is cleared when all write data has been transferred to PCI bus.

Transfer Type Bit 2: (Processor Read)

This bit may be set when the processor initiates a read and is cleared when the read data is returned from the matched data buffer to the processor.

Transfer Type Bit 3: (PCI Read)

This bit may be set when an initiator requests a PCI read cycle and is cleared when PCI read data has been returned to the data buffer.

Transfer Type Bit 4: (Processor Deferred Read)

This bit may be set when the processor initiates a deferred read and is cleared when deferred read data is returned to the processor.

Transfer Type Bit 5: (PCI Deferred Read)

This bit may be set when an initiator requests a PCI deferred read and is cleared when the PCI device returns read data to the matched data buffer.

On page 15, lines 13-26:

Many signals can be used to control communications between the Pentium[[7]]® Pro Processor 41, bridge circuit 40 and PCI device 44. These signals are also used to designate which address (or data) buffer should receive a particular request from the Pentium[[7]]® Pro Processor 41. As can be imagined, it is important for the system to ensure that the proper address is sent to the proper PCI device 44. In addition, because the address and data buffers are separated, the system needs to monitor which address and data buffer has completed its task and is available for more work. The following signals, as listed in Table 3, are used by the internal modules of the bridge circuit 40 to coordinate the movement of information between the modules and by the PCI master controller. Signals that begin with "HS" communicate between the PCI master controller 57 and the CPU slave controller 55. Signals that begin with "HM" communicate between the PCI master controller 57 and the CPU Bus master controller 50. Signals that begin with "PCI" communicate internally between the PCI master controller 57 and a PCI bus interface controller (not shown) which actually controls signals on the PCI bus.

On page 17, lines 9-12 (indents corrected):

The embodiment of the DBA system 60 illustrated in Figure 4 includes an input arbiter 130 that provides control signals to the address buffers 115a-c. The input arbiter 130 interprets the signals described in Table 3, and toggles write enable signals 132a-c that direct the incoming address request 110 into an available buffer.

On page 17, lines 13-19:

As discussed above, the address buffers 115a-c may include three signal paths; one input and two output. The input path may be used to write PCI address transfer requests into the address buffers 115a-c. This may be done when both the HS_REQ and HS_ACK signals are asserted, indicating that the Pentium[[7]]® Pro processor 41 has put

an address request (HS_REQ) on the processor bus 42 and it has been acknowledged (HS_ACK). Once these signals are set, the address and status information is latched into the buffer pointed to by the pointer, top_addr_ptr.

On page 17, lines 20-35 through page 19, line 19 (indents corrected):

For example, when top_addr_ptr points to buffer 115a (e.g.: top_addr_ptr=0) and signals HS_REQ and HS_ACK are asserted (HS_REQ=1; HS_ACK=1), the system may assert a write enable 0 (WE0) signal 132a. This enables the system to write the address and status information into buffer 115a on the next clock cycle. Following a successful write to buffer 115a, top_addr_ptr is incremented by one (top_addr_ptr=1), thereby pointing to buffer 115b. Note that the top_addr_ptr count for the three buffer implementation illustrated in Figure 4 is 0-1-2-0-1-2-0. Through this mechanism, incoming requests are sent to the first available address buffer 115a-c.

The output path corresponding to an address request to read deferred data is determined by the pointer defer_addr_ptr. The defer_addr_ptr will follow the top_addr_ptr until a deferred transfer has been accepted, then it points to the chosen buffer until the deferred data transfer is completed. The defer_addr_prt pointer will then point to the next buffer having a deferred transfer request, if there is one, or begin following the top_addr_ptr pointer again. In most situations, the defer_addr_ptr pointer is incremented to next the deferred transfer or follows top_addr_ptr when read data is returned from the PCI bus to the data buffers (signaled by PCI_DONE) followed by HM_DONE.

The Data Buffers

Referring now to Figure 5, a block diagram of the dynamic buffer allocation system data buffers 200 is shown. The data buffers 200 may be used as illustrated in the embodiment shown in Figure 5, to buffer data transfers between the Pentium processor 41 and PCI bus 43 that are requested by the address buffers 100. As shown, processor write data 205 or PCI read data 210 are inputs to the data buffer scheme 200. Processor write data 205 comes from the processor 41 and is destined for an address corresponding to a particular PCI device 44 on the PCI bus 43. PCI read data 210 is data that has been requested by the processor 41 and is now being sent from the PCI device 44 to the processor 41.

The processor write data 205 and PCI read data 210 act as inputs into a set of input multiplexers 220a-c. These multiplexers are under the control of an input arbiter 240 which uses buffer select signals 242a-c to select the correct one of the Input multiplexers 220 to accept the incoming data stream. This selection process is described more completely below in reference to Figure 6. The input arbiter 240 acts as a selector, activating the proper input multiplexer 220a-c that should receive the incoming data stream based on the particular address buffer that first received the request. In addition, each input multiplexer 220a-c is linked to a single data buffer 250a-c, respectively. Thus, data that is multiplexed by the input multiplexer 220a is sent only to data buffer 250a, while data that is multiplexed by input multiplexer 220b is only sent to data buffer 250b.

As discussed above, the address buffers 115a-c and data buffers 250a-c work together as matched pairs so that, for example, requests placed in address buffer 115a (the first address buffer) will always have their data sent to the first data buffer 250a. The dynamic buffer allocation system address buffers 115a-c (Figure 4) and dynamic buffer allocation system data buffers 250a-c (Figure 5) work in unison through the signals and status bits outlined in Tables 1 and 3 so that an address request into a particular address buffer 115 will always be matched with its appropriate data in a matched data buffer 250. In one embodiment, address buffers 115a, 115b and 115c are matched with data buffers 250a, 250b and 250c, respectively.

The input arbiter 240 asserts write enable signals 252a-c to select when to move data from a particular input multiplexer 220 to its corresponding data buffer 250. Each data buffer can hold up to 255 bits (1 cache line) of data in the embodiment described in Figure 5. However, it should be noted that data buffers having different capacities could be substituted without departing from the spirit of this invention. In addition, each buffer 250 has room for four sets of 8-bit byte enable data wherein each 8-bit byte enable data corresponds to a particular 64-bit segment of data in the buffer.

After data has been placed in one of the data buffers 250a-c, an output arbiter 270 may select an appropriate output multiplexer 275a-c based on the type of request associated with the data held in the data buffer. The data type can be determined by reference to the transfer type bit that is held in the matching address buffer. For example, the output arbiter 270 may provide a CPU select signal 272a to the output multiplexer 275a if the data is to be sent to the processor 41 via the Bus Slave Controller 55 as a

piece of processor read data 290. Alternatively, the output arbiter 270 may provide a PCI select signal 272b to the output multiplexer 275b to send the data from a chosen data buffer to a particular PCI device as a piece of PCI write data 295. Finally, the output arbiter 270 may provide a deferred select signal 272c to the output multiplexer 275c to send deferred data 297 to the processor 41 via the Bus Master Controller 50 of the bridge circuit 40.

In one embodiment, the address/status buffers 115a-c provide the 32-bit addresses for data that are written into their matched data buffers 250a-c. In this manner, the DBA system 60 can match appropriate address requests with the returning data.

The specific signals used within the embodiments described in Figures 4-13 to control the data buffers 250a-c are described in Table 4.

On page 20, lines 8-10 to page 21, lines 1-2 (indents corrected):

The input multiplexers 220a-c are controlled through several pointers, including top_data_ptr, bottom_data_ptr and status signals stored in the address buffers 115a-c. For example if the pointer top_data_ptr = 1 and the transfer type buffer 1 indicates a processor-to-PCI write cycle, then a select signal 242a-c can be asserted to select a particular multiplexer 220a-c that will stroke data into a chosen data buffer 250a-c.

On page 34, lines 23-29:

In addition, the DBA system provides advantages because it is not based on a First In/First Out scheme for managing data flow. For this reason, the system provides more data handling flexibility by allowing higher priority reads and writes to be executed before earlier, lower priority transactions. This is especially important with microprocessors such as the Intel Pentium[[7]]® Pro which may execute many out of order instructions. Because these buffers are not controlled in a first in/first out manner, more flexibility is provided so that the most relevant data is handed off to the bus or processor before less relevant data.